

Python: module vcs.template

vcs.template

[index](#)

Template (P) module

Modules

[Numeric](#)

[RandomArray](#)

[vcs](#)

[copy](#)

[vcs.queries](#)

[vcs](#)

Classes

P

class P

Class: P

Template

Description of P Class:

The template primary method (P) determines the location of each path segment, the space to be allocated to it, and related properties to its display.

Other Useful Functions:

a.show('template')	# Show predefined templates
a.show('texttable')	# Show predefined text table
a.show('textorientation')	# Show predefined text orientation
a.show('line')	# Show predefined line methods
a.listelements('template')	# Show templates as a Python list
a.update()	# Updates the VCS Canvas at regular intervals.
a.mode=1, or 0	# If 1, then automatic update; if 0, then use the update function to update the VCS Canvas.

Example of Use:

a=vcs.init()

To Create a new instance of boxfill use:

box=a.createboxfill('new','quick') # Copies content of 'quick'

box=a.createboxfill('new') # Copies content of 'default'

To Modify an existing boxfill use:

box=a.getboxfill('AMIP_psl')

To Create a new instance of template use:

```

tpl=a.createtemplate('new', 'AMIP')    # Copies content of 'AMIP'
tpl=a.createtemplate('new')             # Copies content of 'default'

To Modify an existing template use:
tpl=a.gettemplate('AMIP')

```

Methods defined here:

```

__init__(self, parent, Pic_name=None, Pic_name_src='default', createP=0)
    #####
    #
    # Initialize the template attributes.
    #
    #####
__setattr__(self, name, value)

drawColorBar(self, colors, levels, legend=None, ext_1='n', ext_2='n', x=None, bg=0, priority=None)
    This function, draws the colorbar, it needs:
    colors : The colors to be plotted
    levels : The levels that each color represent
    legend : To overwrite, saying just draw box at certain values
    ext_1 and ext_2: to draw the arrows
    x : the canvas where to plot it
    bg: background mode ?

drawTicks(self, slab, gm, x, axis, number, vp, wc, bg=0)
    Draws the ticks for the axis x number number
    using the label passed by the graphic method
    vp and wc are from the actual canvas, they have been reset when
    the template is created.

list(self, single=None)
    #####
    #
    # List out template text members (attributes).
    #
    #####
move(self, p, axis)
    move a template by p% along the axis 'x' or 'y'
    positive values of p mean movement toward right/top
    negative values of p mean movement toward left/bottom
    The reference point is t.data.x1/y1
    usage
    t.move(p, axis)
    example:
    t.move(.2, 'x') # move everything right by 20%
    t.move(.2, 'y') # move everything down by 20%

moveto(self, x, y)
    move a template along the axis 'x' or 'y' to p

```

The reference point is t.data.x/y1
 usage
`t.moveto(x,y)`
 example:
`t.moveto(.2,.2) # move everything so that data.x1=.2and data.y1=.2`

`plot(self, slab, gm, bg=0, min=None, max=None)`
 This plots the template stuff on the Canvas, it needs a slab

`ratio(self, Rwished, Rout=None, box_and_ticks=0)`
 Computes ratio to shrink the data area of a template to have
 has the least possible deformation in linear projection

Version: 1.1

Necessary arguments:
`Rwished: Ratio y/x wished`

Optional arguments:
`Rout: Ratio of output (default is US Letter=11./8.5)`
 Also you can pass a string: "A4", "US LETTER", "X"/"SCALING"
`box_and_ticks: Also redefine box and ticks to the new region`

Returned:
`vcs template object`

Usage example:
`## y is twice x`
`t.ratio(2)`

`ratio_linear_projection(self, lon1, lon2, lat1, lat2, Rwished=None, Rout=None, box_and_ticks=0)`
 Computes ratio to shrink the data area of a template in order
 has the least possible deformation in linear projection

Version: 1.1

Notes: Thanks to Karl Taylor for the equation of "optimal" ratios

Necessary arguments:
`lon1, lon2: in degrees_east : Longitude spanned by plot`
`lat1, lat2: in degrees_north : Latitude spanned by plot`

Optional arguments:
`Rwished: Ratio y/x wished, None=automagic`
`Rout: Ratio of output (default is US Letter=11./8.5)`
 Also you can pass a string: "A4", "US LETTER", "X"/"SCALING"
`box_and_ticks: Also redefine box and ticks to the new region`

Returned:
`vcs template object`

Usage example:
`## USA`
`t.ratio linear projection(-135,-50,20,50)`

`reset(self, sub_name, v1, v2, ov1=None, ov2=None)`

this function reset all the attribute who have a sub attribut
also respect how far from original position you are
i.e you move to `x1,x2` from `old_x1, old_x2`
if your current `x1` value is not == to `old_x1_value`, then resp
usage:
`reset(sub_name,v1,v2,ov1=None,ov2=None)`

Example:
`t.reset('x',x1,x2,t.data.x1,t.data.x2)`
#where t is a vcs template object

scale(self, scale, axis='xy', font=-1)
scale a template along the axis 'x' or 'y' by scale
positive values of p mean increase
negative values of p mean decrease
The reference point is `t.data.x/y1`
usage
`t.scale(scale,axis='xy',font=-1)`
scale: any number
axis can be 'x', 'y' or 'xy' (which means both)
font can be 1/0/-1
0: means do not scale the fonts
1: means scale the fonts
-1: means do not scale the fonts unless axis='xy'

Example:
`x=vcs.init()`
`t=x.createtemplate('a_template')`
`t.scale(.5) # halves the template size`
`t.scale(1.2) # upsize everything to 20% more than the original`
`t.scale(2,'x') # double the x axis`

reference is `t.data.x1/y1`

scalefont(self, scale)
Scales the tempate font by scale
Usage:
`scalefont(scale)`

Example:
`x=vcs.init()`
`t=x.createtemplate('a_template')`
`t.scalefont(.5) # reduces the fonts size by 2`

script(self, script_filename=None, mode=None)
Function: `script` # Calls `_vcs.s`

Description of Function:
Saves out a template object in VCS or Python script for
file.

Example of Use:
`script(scriptfile_name, mode)`
where: `scriptfile_name` is the output name of the

mode is either "w" for replace or "a" for append.

Note: If the the filename has a ".py" at the end of it, then a Python script will be produced. If the filename has a ".scr" extension, then a VCS script will be produced. If neither extension is present, then by default a Python script will be produced.

```
a=vcs.init()
templt=a.createtemplate('temp')      # create a template object
templt.script('filename.py')         # Append to a Python file
templt.script('filename.scr')        # Append to a VCS file
templt.script('filename','w')         # Create or overwrite template
```

Functions

getPmember(self, member)

```
#####
#
# Function:      getPmember
#
# Description of Function:
#     Private function that retrieves the template members from
#     structure and passes it back to Python.
#
#
# Example of Use:
#     return_value =
#     getPmember(self, name)
#             where: self is the class (e.g., P)
#                     name is the name of the member that is being
#     #
#####
#
```

renameP(self, old_name, new_name)

```
#####
#
# Function:      renameP
#
# Description of Function:
#     Private function that renames the name of an existing template
#     graphics method.
#
#
# Example of Use:
#     renameP(old_name, new_name)
#             where: old_name is the current name of template graphics
#                     new_name is the new name for the template graphics
#     #
#####
#
```

```
setPmember(self, member, value)
#####
#
# Function:      setPmember
#
# Description of Function:
#      Private function to update the VCS canvas plot. If the can
#      set to 0, then this function does nothing.
#
#
# Example of Use:
#      setPmember(self, name, value)
#          where: self is the class (e.g., P)
#                  name is the name of the member that is being
#                  value is the new value of the member (or att
#
#####
#####
```

Data

StringTypes = (<type 'str'>, <type 'unicode'>)